ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

(Laboratorio di)
Amministrazione di sistemi

Primi passi sulla riga di comando

Marco Prandini

Dipartimento di Informatica – Scienza e Ingegneria

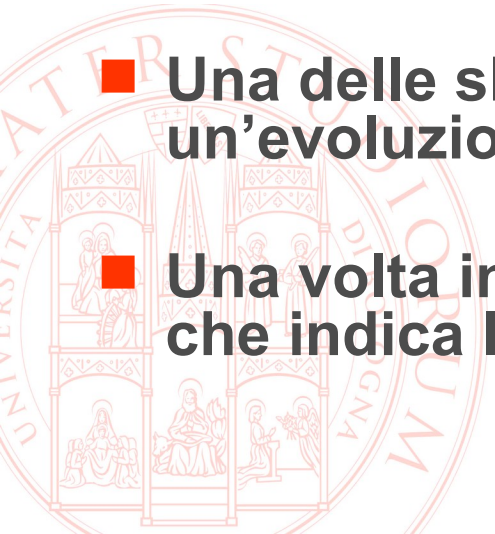
Login e interprete dei comandi

- Lo strumento più potente,flessibile e soprattutto più standard con cui amministrare un sistema Unix è l'interfaccia testuale a **riga di comando**.

- Componente essenziale è la **shell** o interprete dei comandi, che permette di svolgere compiti quali job control, mette a disposizione variabili e strutture di controllo per scrivere semplici programmi, e permette di invocare gli eseguibili installati nel sistema.
  - Possiamo definire shell qualsiasi interfaccia tra utente e OS

- Una delle shell più usate è **bash**, (Bourne-again shell) un'evoluzione della classica Bourne Shell di Unix.

- Una volta inserite le credenziali, il sistema presenta un *prompt* che indica la disponibilità dell'interprete ad accettare comandi



Chi sono, chi c'è

- Poiché è del tutto comune disporre di differenti identificativi utente con cui lavorare, è utile disporre di un comando per sapere quale e' l'identificativo con il quale si sta operando:
  - `whoami` indica il proprio username
  - `id` dà informazioni sull'identità e sul gruppo di appartenenza
  - `who` indica chi e' attualmente collegato alla macchina

Esecuzione dei comandi

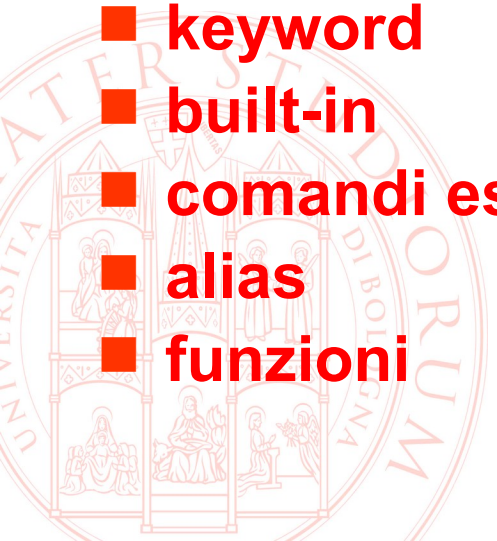
La shell indica il proprio stato di 'pronto' con una stringa di caratteri visualizzati nella parte iniziale della prima linea vuota. Questa stringa e' detta 'prompt'.

I caratteri digitati dall'utente dopo il prompt e terminati dal fine linea (ritorno a capo) costituiscono la command line.

Questa, tipicamente, contiene comandi e argomenti.

I comandi digitati sulla command line possono essere:

- keyword
- built-in
- comandi esterni
- alias
- funzioni



Tipi di comandi

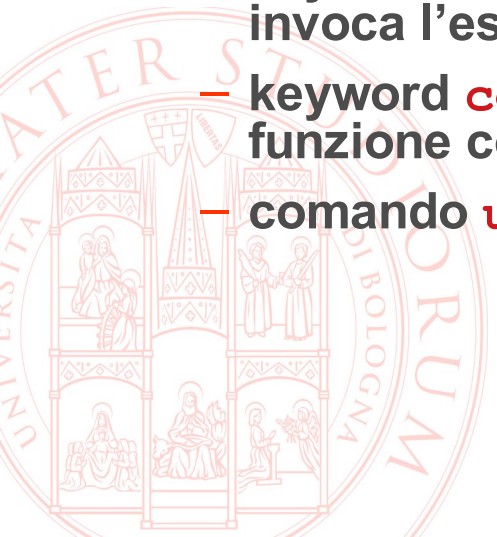
- Una **keyword** è un comando che ha il compito di modificare l'esecuzione di altri comandi (es. misurare il tempo di esecuzione, innescare un ciclo)
- Un built-in e' un comando **direttamente eseguito dal codice interno della shell** che lo interpreta e lo 'converte' in azioni sul sistema operativo. Un esempio tipico è costituito dai comandi per la navigazione del filesystem.
- Un comando esterno e' un **file eseguibile che viene localizzato e messo in esecuzione dalla shell** (tipicamente in un processo figlio). La shell può attenderne o meno la conclusione prima di accettare nuovi comandi.
- Un **alias** è una stringa che viene sostituita da un'altra
- Una **funzione** è un'intera sequenza di comandi shell, con un nome, a cui possono essere passati parametri

Ricerca dei comandi tra i diversi tipi

- Come distinguere quale tipo di comando viene eseguito?
  - comando `type`

```
$ type -a echo
echo is a shell builtin
echo is /bin/echo
```

- Come alterare l'ordine di default?
  - backslash `\` davanti al comando: previene solo l'espansione degli alias
  - keyword `builtin`: previene l'espansione degli alias e l'uso di funzioni e invoca l'esecuzione del builtin specificato, se esiste
  - keyword `command`: utilizza un comando esterno anche se esiste una funzione con lo stesso nome
  - comando `unalias`: cancella un alias definito in precedenza



Alias

- La shell mette a disposizione alcuni sistemi per memorizzare linee di comando complesse in comandi più semplici da invocare. Uno di questi è l'**alias** che permette di attribuire un nome ad una command line:

```
alias miols='ls -l'
```

- Le associazioni definite con alias vanno perse al termine della sessione, vedremo come renderle persistenti.

Alias vs. builtin e comandi

- Una volta definiti, gli alias (o come descritto in seguito, le funzioni) questi hanno la priorità rispetto ai builtin ed i comandi omonimi.
- Per analizzare e pilotare la configurazione, sono utili i comandi `type`, `builtin`, `command` e `unalias`

```
$ alias echo='echo ~~~'
$ echo test
~~~ test
$ \echo test
test
$ builtin echo test
test
$ type echo
echo is aliased to `echo ~~~'
$ unalias echo
$ type -a echo
echo is a shell builtin
echo is /bin/echo
```

`\` previene solo l'espansione degli alias,
se `echo` fosse stato rifedinito come funzione
sarebbe stata usata quest'ultima

`builtin` invece fa override sia delle definizioni
di alias che di funzioni;
vale solo per riaccedere ai builtin:
se è stato ridefinito un comando esterno,
si usi `command` per invocare la versione originale

Ricerca dei comandi esterni

- Per lanciare un eseguibile lo si può individuare col percorso completo
  - assoluto

    ```
    /usr/local/bin/top
    ```
  - relativo

    ```
    ./mycommand
    ```
- Tra le variabili d'ambiente comuni, la shell utilizza PATH per eseguire la ricerca dei comandi nel file system. La sua struttura è quella di un elenco di directory separate da :

  ```
  PATH=/bin:/usr/bin:/sbin
  ```
- Nel caso di più eseguibili omonimi in directory diverse?
  - lista ordinata, il sistema usa la prima istanza che trova
  - `which` Permette di sapere quale versione si sta usando:

    ```
    # which passwd
    /usr/bin/passwd
    ```
- Per consentire automaticamente l'esecuzione di programmi presenti nella directory corrente la variabile PATH deve contenere la directory .
  - Non è una buona norma, è facile lanciare per distrazione comandi errati
  - Meglio usare il percorso esplicito (vedi inizio slide)

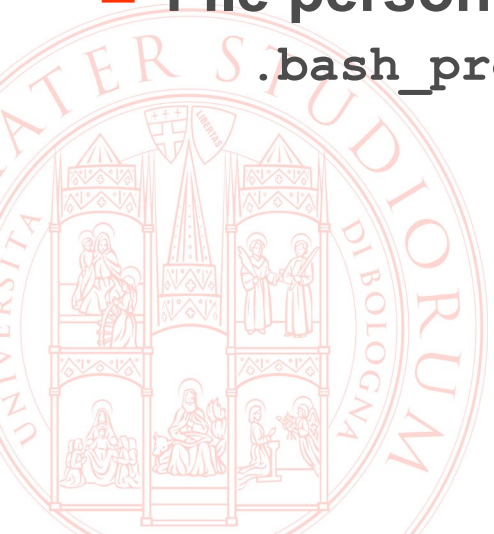
Impostazioni di default

- Per ottenere automaticamente all'avvio della shell
  - assegnazione di valori a variabili come PATH
  - definizione di alias di comandi
  - … esecuzione di qualsiasi comando shell utile per inizializzare l'ambiente

  si ricorre ai file di configurazione di bash
- Si vedano (quasi all'inizio) la sezione INVOCATION e (quasi in fondo) la sezione FILES di *man bash*
- File globali

  `/etc/profile` `/etc/bash.bashrc`
- File personali (nella home)

  `.bash_profile` `.bash_login` `.profile` `.bashrc`



Documentazione dei comandi (e non solo)

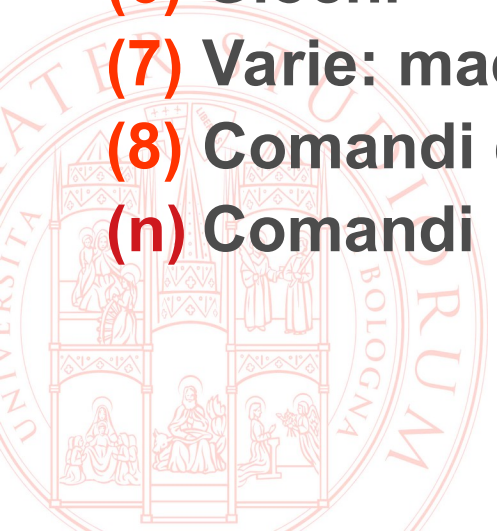
- **man pages** – ogni applicazione installa "pagine di manuale" relative al suo utilizzo e configurazione.
  - Le man pages si leggono con il comando *man*
  - I builtin, non essendo programmi installati indipendentemente, non hanno man page.
    - Un sommario del loro funzionamento può essere visualizzato con

      help <builtin>
    - Inoltre, naturalmente, sono documentati nella man page `bash(1)`
- **info files** – a metà strada tra la man page e l'ipertesto, si leggono con il comando *info*, che invoca l'editor emacs appositamente esteso per gestire questi file
- **HOWTO** – documenti specifici per la risoluzione dei più svariati problemi pratici, sono raccolti in un pacchetto e vengono tutti installati in
  /usr/[share/]doc/HOWTO
  Inoltre, sotto */usr/doc/HOWTO/translations/it* si possono trovare la maggior parte degli HOWTO tradotti in italiano.
- **on-line** – troppe fonti per citarle... un punto di partenza può essere http://tldp.org/ The Linux Documentation Project

Categorie di man pages

Una installazione standard di unix mette a disposizione innumerevoli pagine di manuale raggruppate in sezioni:

(1) User commands
(2) Chiamate al sistema operativo
(3) Funzioni di libreria, ed in particolare
(4) File speciali (/dev/*)
(5) Formati dei file, dei protocolli, e delle relative strutture C
(6) Giochi
(7) Varie: macro, header, filesystem, concetti generali
(8) Comandi di amministrazione riservati a *root*
(n) Comandi predefiniti del linguaggio Tcl/Tk



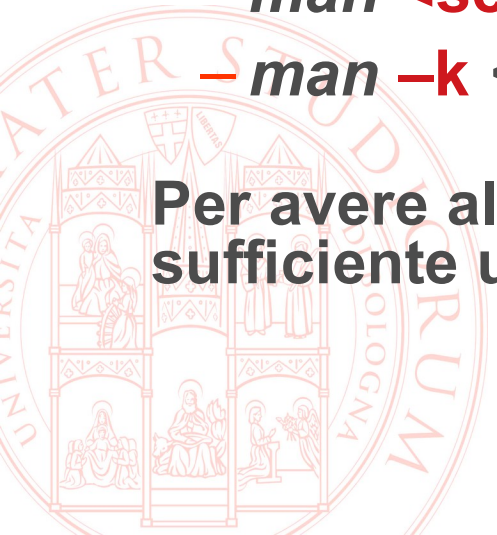
Accesso alle man pages

- L'accesso alle pagine si ottiene con il comando
- man <nome della pagina>

- Spesso il nome della pagina coincide con il comando o il nome del file di configurazione che essa documenta.

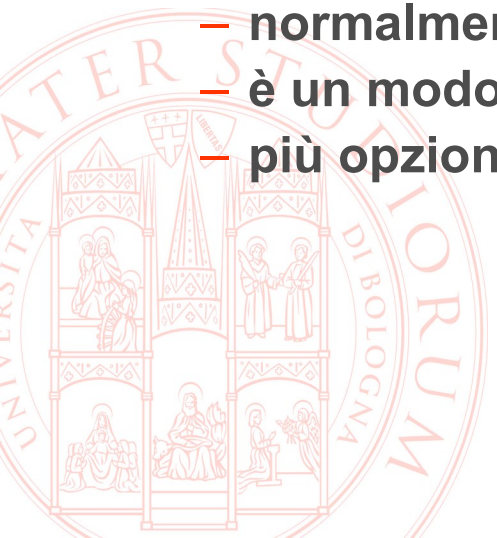
- Alcune opzioni utili sono qui riassunte:
  - *man –a <comando>* cercherà in tutte le sezioni
  - *man <sez.> <comando>* cercherà nella sezione specificata
  - *man –k <keyword>* cercherà tutte le pagine attinenti alla parola chiave specificata

Per avere altre informazioni sul comando man è ovviamente sufficiente usare man man.



Gli argomenti

- Ogni comando, sia built-in che esterno, può accedere ai caratteri che seguono la propria invocazione sulla command line.
  - La shell inserisce in memoria l'ARGV prima di generare il processo
- I gruppi di caratteri, **separati da spazi**, rappresentano gli **argomenti**, cioè i dati su cui si vuole che il comando operi.
- Un argomento che inizia con il carattere '-' è chiamato solitamente *opzione*
  - normalmente non è un vero e proprio dato da elaborare
  - è un modo di specificare una variante al comportamento del comando
  - più opzioni possono essere solitamente raggruppate in un'unica stringa.



Esempi di argomenti e opzioni

ls /home	arg #1="/home" indica la directory di cui elencare il contenuto
ls -l /home	arg #1=opzione l indica che si desidera un listato in forma “lunga”
ls -l -a /home/alex	arg #1 e #2 = opzioni l (lunga) ed a (all)
ls -la /home/alex	arg #1 = opzioni concatenate l+a (identico a prima)

Digitare interattivamente la command line

- Iniziando a scrivere un comando (come primo elemento) o un nome di file (come argomento), e battendo TAB, bash completa automaticamente la stringa se non ci sono ambiguità, o suggerisce come completarla correttamente.

  Es.: al prompt digito `pass`<TAB> → compare `passwd` seguito da spazio ad indicare che passwd è l'unico completamento possibile

- Se ci sono ambiguità, una seconda pressione di <TAB> mostra i completamenti possibili.

  Es.
  - digito `ls /etc/pam`<TAB> → compare `/etc/pam.`
  - digito <TAB> → vengono elencati `pam.conf` e `pam.d/`
  - aggiungo "`c`" e digito <TAB> → compare `/etc/pam.conf`

Richiamare comandi passati

- `history` mostra l'elenco di tutti i comandi eseguiti in un terminale. Per richiamarli sulla command line, basta usare la freccia-su, appariranno (editabili) dal più recente al più vecchio
- La history è anche ricercabile interattivamente: al prompt basta digitare CTRL-r per far apparire un prompt (reverse-i-search); digitando una stringa, verrà mostrato il comando più recente che la contiene.
- Per navigare verso comandi impartiti precedentemente basta digitare nuovamente CTRL-r.
- Individuato il comando desiderato, si può lanciare direttamente premendo invio, o renderlo editabile sulla command line con freccia-destra o freccia-sinistra

Documentare le attività svolte sulla shell

- Il comando `script` permette di catturare in un file una sessione di lavoro al terminale, esattamente come compare a video, quindi sia i comandi impartiti che il loro risultato.

- Per terminare l'attività, digitare `exit` o premere CTRL-d

Brevissima introduzione all'editor VIM

- VIM e' un editor a 'tutto schermo', versione più amichevole dello storico VI.
- Utilizza una interfaccia 'modale'. In altre parole, il programma si puo trovare in uno dei seguenti stati:
  - COMMAND
    - Il cursore è posizionato sul testo
    - la tastiera è utilizzabile solo per richiedere l'esecuzione di comandi, e non per introdurre testo.
    - I caratteri digitati non vengono visualizzati.
  - INPUT
    - Tutti i caratteri digitati vengono visualizzati ed inseriti nel testo.
  - DIRECTIVE
    - Ci si trova posizionati con il cursore nella linea direttive (l'ultima linea del video) e si possono richiedere tutti i comandi per il controllo del file.

Passaggi di stato

- I passaggi di stato avvengono digitando uno dei seguenti caratteri:

COMMAND MODE	→	INPUT MODE	`oOiIaACR`
INPUT MODE	→	COMMAND MODE	`<ESC>`
COMMAND MODE	→	DIRECTIVE MODE	`:/?`
DIRECTIVE MODE	→	COMMAND MODE	`<RET>`

Gestione file

- I comandi per caricare/salvare/uscire sono DIRECTIVE
  - `:r <nome>` inserisce il contenuto del file `nome` al punto del cursore
  - `:w` scrive il file corrente
  - `:q` esce (`:q!` se si vuole uscire senza salvare)
  - `ZZ` scrive ed esce

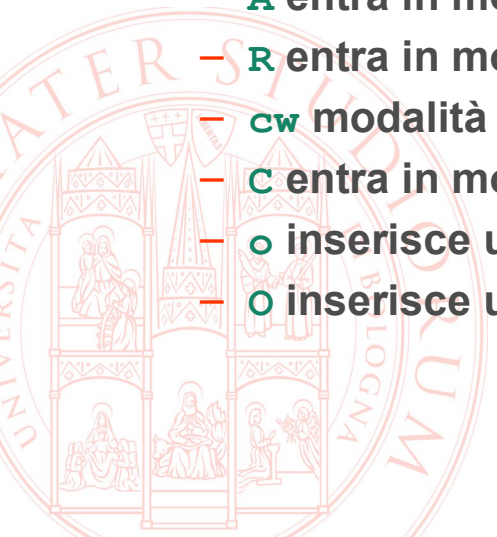
Comandi di spostamento

- In COMMAND MODE, e' possibile richiedere lo spostamento del cursore con i comandi di movimento:
  - `h` 1 spazio a sinistra (come backspace)
  - `l` 1 spazio a destra (come space)
  - `k` 1 linea sopra (stessa colonna)
  - `j` 1 linea sotto (stessa colonna)

  (in sostituzione di questi quattro caratteri, si possono anche utilizzare le frecce per spostarsi nelle diverse direzioni)
- Inizio/fine riga
  - `^` posizionamento all'inizio della riga
  - `$` posizionamento alla fine della riga
- Inizio/fine file
  - `gg` posizionamento sulla prima linea del testo
  - `G` posizionamento sull'ultima linea del testo
  - `#G` posizionamento sulla linea numero "#"
    - Esempio: `3G` `5G` `175G`
    - Notate che i comandi non vengono visualizzati!

Modifica

- In COMMAND MODE è anche possibile apportare modifiche
- cancellazione
  - `x` cancella il carattere su cui si trova il cursore
  - `dd` cancella la linea su cui si trova il cursore
- modifica di singoli elementi carattere (non si passa a INPUT MODE)
  - `rX` rimpiazza il carattere sotto il cursore con `X`
- modifica (provocano l'ingresso in INPUT MODE):
  - `i` entra in modalità 'inserimento' nella posizione del cursore
  - `I` entra in modalità 'inserimento' a inizio riga
  - `a` entra in modalità 'append' nella posizione del cursore
  - `A` entra in modalità 'append' a fine riga
  - `R` entra in modalità ‘replace’ (sovrascrittura)
  - `cw` modalità ‘change word’, elimina la parola che inizia sotto il cursore ed edita
  - `C` entra in modalità ‘change’ (elimina fino a fine riga)
  - `o` inserisce una linea vuota sotto al cursore
  - `O` inserisce una linea vuota sopra al cursore



Ricerca e sostituzione

- Digitando la barra, si entra in DIRECTIVE MODE per cercare stringhe
  - es. `/ciao<RET>` posiziona il cursore sulla prima occorrenza successiva della stringa ciao (se esiste)
  - Digitando `n` si passa alla successiva (next)
  - Digitando `N` si passa alla precedente
- Se al posto di `/` si usa `?`, la direzione di ricerca è verso l'inizio (e i significati di `n` e `N` si adeguano di conseguenza)
- `/<RET>` e `?<RET>` ripetono l'ultima ricerca
- `:s/trova/sostituisci/`
  - cerca 'trova' nella linea corrente e lo sostituisce con 'sostituisci'
- `:%s/trova/sostituisci/cgi`
  - cerca 'trova' in ogni linea del file e lo sostituisce con 'sostituisci'
  - dopo aver chiesto conferma (`c`),
  - anche più volte nella stessa linea (`g`),
  - case-insensitive (`i`)
- `%` è una scorciatoia per `1,$` - in realtà il comando può essere invocato come `:I,Fs/.../.../` per applicarlo alle linee tra I e F

Combinazioni e ripetizioni

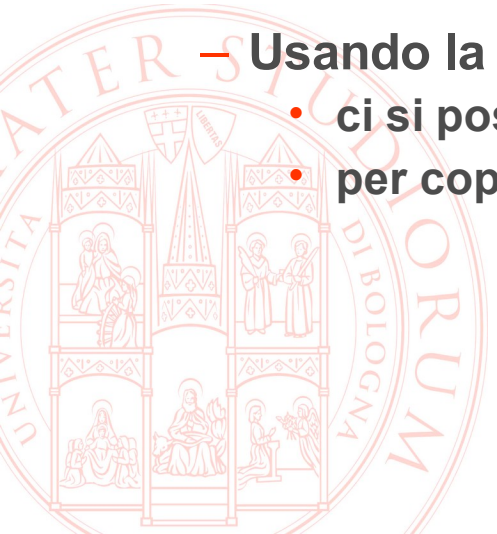
- La ricerca e gli spostamenti possono essere usati come "terminatore" per alcuni comandi di modifica, ad esempio:
  - `d$` cancella dalla posizione corrente a fine riga
  - `dG` cancella dalla posizione corrente a fine file
  - `c/ciao<RET>` cancella dalla posizione corrente alla prima occorrenza della stringa ciao e si porta in insert mode
- Digitando il punto `.` si ripete l'ultimo comando impartito
- Facendo precedere un numero N a un comando, il comando viene eseguito N volte consecutivamente
  - Es. `10x` cancella 10 caratteri
- Digitando `u` (undo) si annulla l'ultima azione eseguita

Copia & incolla (1)

- i comandi di copia utilizzano dei buffer interni a vi; tipicamente si utilizza il buffer standard, ma e' possibile specificarne altri:
  - `yy` copia la linea corrente nel buffer
  - `"ayy` copia la linea corrente nel buffer "a"
- Il comando `d` esegue il cut anziché il copy
- Il paste (incolla) si ottiene con i comandi:
  - `p` incolla dopo la linea corrente
  - `P` incolla prima della linea corrente

Copia & incolla (2)

- per copiare blocchi di linee si può procedere in diversi modi
  - Usando marcatori
    - ci si posiziona sulla prima linea del blocco
    - `ma` marca la posizione con il simbolo "a"
    - ci si posizione sull'ultima linea del blocco
    - `y'a` copia nel buffer tutto il testo dalla posizione marcata "a" in precedenza fino alla posizione corrente
  - Usando le ripetizioni
    - ci si posiziona sulla prima linea del blocco
    - per copiare 10 righe si digita `10yy`
  - Usando la ricerca
    - ci si posiziona sulla prima linea del blocco
    - per copiare fino alla parola 'basta' (esclusa) si digita `y/basta<RET>`



Macro

- Digitando `q<lettera minuscola>` (es: `qa`) inizia la registrazione della macro identificata dalla lettera specificata.
- Tutte le azioni compiute saranno registrate, finché non si preme nuovamente `q` per terminare la registrazione.
- Digitando `@<lettera minuscola>` viene riprodotta la macro.
  - (naturalmente si può riprodurre N volte digitando `N@<lettera>`)